МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе № 2-2

по дисциплине «Программирование»

Тема: «Использование коллекций»

Выполнил: Меньков С. А., МВА-18

Проверил: Самойлова Т. А.

Москва 2019

# **Цель работы**

Изучить основные интерфейсы коллекций и классы их реализации, методы создания и использования коллекций для решения прикладных задач. Изучить основные принципы работы с файлами.

**Задание**

Обработать данные анализа сырья

|  |  |
| --- | --- |
| Анализ загрязненности сырья | |
| Сырье | Загрязненность, % |
| (до 20 символов) | 99,9 |

Вычислить среднюю загрязненность и вывести перечень сырья с превышением этой величины «Сырье», загрязненность

**Текст программы**

public class Main {  
  
 public static void main(String[] args) {  
  
 System.out.println("Reading data from file ...");  
 Storage storage = new Storage("fileDB.txt");  
 System.out.println("Source Data Table:");  
 storage.printTable(storage.getSourceList());  
 int middlePollution = storage.getMiddlePollution();  
 List<Pair> excessPollutionList = storage.getExcessPollutionList(middlePollution);  
 System.out.println("Excess Pollution Data Table:");  
 storage.printTable(excessPollutionList);  
 System.out.println("Saving result into the file ...");  
 storage.saveData("ExcessPollutions.txt", excessPollutionList);  
  
 }  
}

public class Pair {  
  
 private String rawMat;  
 private double pollution;  
  
  
 public Pair(String rawMat, double pollution) {  
 this.rawMat = rawMat;  
 this.pollution = pollution;  
 }  
  
 public String getRawMat() {  
 return rawMat;  
 }  
  
 public double getPollution() {  
 return pollution;  
 }  
  
}

public class Storage {  
  
 private static final String ENTITY = "Сырьё";  
 private static final String VALUE = "Загрязненность, %";  
  
 private final File sourceFile;  
 private final List<Pair> sourceList;  
  
 private int maxEntityLength;  
 private int maxValueLength;  
  
 private Storage() {  
 sourceFile = null;  
 sourceList = null;  
 // DB can`t be empty  
 }  
  
 public Storage(String path) {  
 sourceFile = new File(path);  
 sourceList = new ArrayList<>();  
 try {  
 readData();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void readData() throws FileNotFoundException {  
 Scanner sc = new Scanner(sourceFile);  
 String[] line;  
 int maxEntityLength = ENTITY.length();  
 int maxValueLength = VALUE.length();  
 while (sc.hasNextLine()) {  
 line = sc.nextLine().split(",");  
 if (line.length == 2) {  
 if (maxEntityLength < line[0].length())  
 maxEntityLength = line[0].length();  
 if (maxValueLength < line[1].length())  
 maxValueLength = line[1].length();  
 String rawMat = null;  
 double pollution = 0;  
 try {  
 rawMat = line[0];  
 pollution = Double.parseDouble(line[1].trim());  
 } catch (NumberFormatException ex) {  
 System.out.println("Error while parsing value from source file !!!");  
 System.exit(1);  
 }  
 sourceList.add(new Pair(rawMat, pollution));  
 } else {  
 System.out.println("Error in the source file format <String, int> !!!");  
 System.exit(1);  
 }  
 }  
 this.maxEntityLength = maxEntityLength;  
 this.maxValueLength = maxValueLength;  
 sc.close();  
 }  
  
 public void saveData(String path, List<Pair> data) {  
 File outputF = new File(path);  
 FileWriter writer;  
 try {  
 if (outputF.exists() || outputF.createNewFile()) {  
 writer = new FileWriter(outputF, false);  
 for (Pair p : data) {  
 writer.write(p.getRawMat() + "," + p.getPollution() + "\n");  
 }  
 writer.close();  
 }  
 } catch (IOException e) {  
 System.out.println("Error while saving changes !!!");  
 }  
 }  
  
 public List<Pair> getSourceList() {  
 return sourceList;  
 }  
  
 public List<Pair> getExcessPollutionList(int maxAllowPollution) {  
 ArrayList<Pair> excessPollutionList = new ArrayList<>();  
 for (Pair p : sourceList) {  
 if (p.getPollution() > maxAllowPollution) {  
 excessPollutionList.add(p);  
 }  
 }  
 return excessPollutionList;  
 }  
  
 public int getMiddlePollution() {  
 int middlePollution = 0;  
 for (Pair p : sourceList) {  
 middlePollution += p.getPollution();  
 }  
 middlePollution /= sourceList.size();  
  
 System.out.println("The middle pollution is: " + middlePollution);  
  
 return middlePollution;  
 }  
  
  
 private void appendSplitter(StringBuilder table) {  
 table.append('+');  
 for (int i = 0; i < maxEntityLength; i++)  
 table.append('-');  
 table.append('+');  
 for (int i = 0; i < maxValueLength; i++) {  
 table.append('-');  
 }  
 table.append("+\n");  
 }  
  
 public void printTable(List<Pair> dataList) {  
 StringBuilder table = new StringBuilder();  
 appendSplitter(table);  
 ///Header///  
 table.append('|');  
 int padding = maxEntityLength - ENTITY.length() > 0 ? (maxEntityLength - ENTITY.length()) / 2 : 0;  
 int cursor;  
 for (cursor = 0; cursor < padding; cursor++) {  
 table.append(' ');  
 }  
 table.append(ENTITY);  
 cursor += ENTITY.length();  
 for (; cursor < maxEntityLength; cursor++) {  
 table.append(' ');  
 }  
 table.append('|');  
 padding = maxValueLength - VALUE.length() > 0 ? (maxValueLength - VALUE.length()) / 2 : 0;  
 for (cursor = 0; cursor < padding; cursor++) {  
 table.append(' ');  
 }  
 table.append(VALUE);  
 cursor += VALUE.length();  
 for (; cursor < maxValueLength; cursor++) {  
 table.append(' ');  
 }  
 table.append("|\n");  
 ///////////  
  
 appendSplitter(table);  
  
 /// data ///  
 table.append('|');  
 String rawMat;  
 String pollution;  
 for (Pair p : dataList) {  
 ///////////////////////////////////////////  
 rawMat = p.getRawMat();  
 pollution = String.valueOf(p.getPollution());  
 padding = maxEntityLength - rawMat.length();  
 for (int i = 0; i < padding; i++) {  
 table.append(' ');  
 }  
 table.append(rawMat);  
 table.append('|');  
 padding = maxValueLength - pollution.length();  
 for (int i = 0; i < padding; i++) {  
 table.append(' ');  
 }  
 table.append(pollution);  
 table.append('|');  
 table.append("\n|");  
 }  
 table.deleteCharAt(table.length() - 1);  
 ////////////  
 appendSplitter(table);  
  
 System.out.println(table);  
 }  
   
}

**Результат выполнения**

Исходный файл (рис. 1)

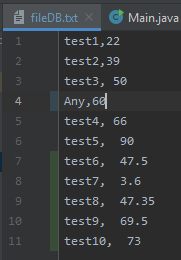


Рис. 1

Вывод программы (рис. 2)

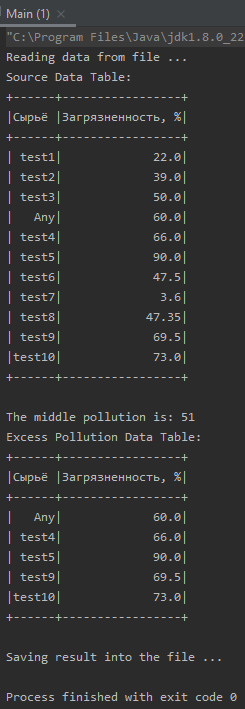


Рис. 2

В результате был создан новый файл и заполнен данными (рис. 3)

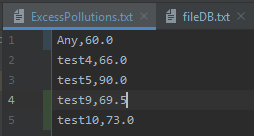


Рис. 3